Final Technical Report (2/1/95-3/31/96)

AFOSR Grant No. F49620-95-1-0126

'DEVELOPMENT OF AN ADVANCED IMPLICIT ALGORITHM FOR MHD COMPUTATIONS ON PARALLEL SUPERCOMPUTERS"

Submitted to

Air Force Office of Scientific Research
Bolling AF Base
Washington, DC 20332-8080

UNIVERSITY OF WASHINGTON
Aerospace & Energetics Research Program, Box 352250
Seattle, Washington 98195-2250

Dr. Uri Shumlak
Principal Investigator

May 31, 1996

unclassified/unlimited
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION unclassified/unlimited	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT unclassified/unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Aerospace & Energetics Research Program 6b. OFFICE SYMBOL (If applicable) 2250	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research (ONR)
6c. ADDRESS (City, State and ZIP Code) University of Washington Grant & Contract Services 3935 Univ Way NE, Seattle, WA 98105-6613	7b. ADDRESS (City, State and ZIP Code) 1107 NE 45th St., Suite 350 Seattle, WA 98105-4631
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFOSR 8b. OFFICE SYMBOL (If applicable) /NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F49620-95-1-0126
8c. ADDRESS (City, State and ZIP Code) 110 Duncan Avenue, Suite B115 Bolling AFB, DC 20332-8080	10. SOURCE OF FUNDING NOS.

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO
	2304	/CS	

11. TITLE (Include Security Classification) "DEVELOPMENT OF AN ADVANCED IMPLICIT ALGORITHM FOR MHD COMPUTATIONS ON PARALLEL SUPERCOMPUTERS"

12. PERSONAL AUTHOR(S)
SHUMLAK, Uri

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
final technical report	FROM 95/2/1 TO 96/3/31	96/5/31	56

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
implicit algorithm, parallel computer, lower-upper symmetric-Gauss-Siedel, LUSGS, approximate Riemann solver, magnetohydrodynamic, MHD, Hartmann flow

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The primary objective of this project is to develop an advanced algorithm for parallel supercomputers to model time-dependent magnetohydrodynamics (MHD) in all three dimensions. This will provide a valuable tool for the design and testing of plasma related technologies that are important to the Air Force and industry. These applications include nuclear weapons effects simulations, radiation production for counter proliferation, fusion for power generation, and advanced plasma thrusters for space propulsion.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED [✓] SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION unclassified/unlimited

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Donald W. Allen, Director G&C Services	206 543-4043	5754

DD FORM 1473, 83 APR   EDITION OF 1 JAN 73 IS OBSOLETE.   unclassified/unlimited
SECURITY CLASSIFICATION OF THIS PAGE

Development of an Advanced Implicit Algorithm for MHD Computations on Parallel Supercomputers

Contract Number: F49620-95-1-0126

U. Shumlak
Department of Aeronautics and Astronautics, Box 352250
University of Washington, Seattle, WA 98195-2250

Contents

List of Figures

1 Executive Summary

The primary objective of this project is to develop an advanced algorithm for parallel supercomputers to model time-dependent magnetohydrodynamics (MHD) in all three dimensions. This will provide a valuable tool for the design and testing of plasma related technologies that are important to the Air Force and industry. Implementing the algorithm on parallel supercomputers will allow the detailed modeling of realistic plasmas in complex three-dimensional geometries.

We have developed a time-dependent, two-dimensional, arbitrary-geometry version of the algorithm, placed it into a testbed code, added the modifications necessary for viscous and resistive effects, and tested the code against known analytical problems. We have implemented the algorithm on a parallel architecture and investigated parallelization strategies. Future plans include installing the algorithm into MACH2, optimizing the parallelization, extending the code to three dimensions, installing the three dimensional algorithm into MACH3,[1] and calibrating the code with experimental data.

As a result of this project several professional collaborations now exist between the Department of Aeronautics and Astronautics at the University of Washington and the Air Force Phillips Laboratory, Lawrence Livermore National Laboratory, the University of Michigan, the University of Colorado, and other departments at the University of Washington. The work from this project has been presented at international conferences and one publication has already been published in a refereed journal and another publication has been accepted for publication pending revisions.

2 Project Description

Plasmas are essential to many technologies that are important to the Air Force, some of which have dual-use potential. These applications include nuclear weapons effects simulations, radiation production for counter proliferation, fusion for power generation, and advanced plasma thrusters for space propulsion. In general, plasmas fall into a density regime where they exhibit both collective (fluid) behavior and individual (particle) behavior. Many plasmas of interest can be modeled by treating the plasma like a conducting fluid and assigning macroscopic parameters that accurately describe its particle-like interactions. The magnetohydrodynamic (MHD) model is a plasma model of this type.

2.1 Research Objectives

The objectives of the project are to:

- Develop a coupled, implicit, time-accurate algorithm for three-dimensional, viscous, resistive MHD simulations;
- Incorporate the algorithm into the MACH3 code, which was developed at the Air Force Phillips Laboratory;
- Validate the code with analytical and experimental data; and
- Apply the code to analyze plasma experiments at the University of Washington [Helicity Injected Tokamak (HIT)[2]] and at the Phillips Laboratory [the liner implosion system (WFX),[3] the dense plasma focus experiment, and magnetic flux compression generators].

2.2 Technical Description

2.2.1 MHD Plasma Model

The three-dimensional, viscous, resistive MHD plasma model is a set of mixed hyperbolic and parabolic equations. The Navier-Stokes equations are also of this type. This project applies some advances that have been made in implicit algorithms for the Navier-Stokes equations to the MHD equations. These implicit algorithms solve the equation set in a fully coupled manner, which generates better accuracy than the current methods used for MHD simulations.

When expressed in conservative, non-dimensional form, the MHD model is described by the following equation set.

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho\mathbf{v}\\ \mathbf{B}\\ e\end{bmatrix} + \nabla\cdot\begin{bmatrix}\rho\mathbf{v}\\ \rho\mathbf{v}\mathbf{v}-\mathbf{B}\mathbf{B}+(p+\mathbf{B}\cdot\mathbf{B}/2)\,\bar{\bar{\mathrm{I}}}\\ \mathbf{v}\mathbf{B}-\mathbf{B}\mathbf{v}\\ (e+p+\mathbf{B}\cdot\mathbf{B}/2)\,\mathbf{v}-(\mathbf{B}\cdot\mathbf{v})\,\mathbf{B}\end{bmatrix} =$$

$$\nabla\cdot\begin{bmatrix}0\\ (ReAl)^{-1}\,\bar{\bar{\tau}}\\ (RmAl)^{-1}\,\bar{\bar{\Xi}}(\bar{\bar{\eta}},\mathbf{B})\\ (ReAl)^{-1}\,\mathbf{v}\cdot\bar{\bar{\tau}}-(RmAl)^{-1}\,\bar{\bar{\eta}}\cdot(\nabla\times\mathbf{B})\times\mathbf{B}+\frac{M_i}{2}\,(PeAl)^{-1}\,\bar{\bar{k}}\cdot\nabla T\end{bmatrix} \quad (1)$$

The variables are density (ρ), velocity ($\mathbf{v}$), magnetic induction ($\mathbf{B}$), pressure (p), energy density (e), and temperature (T). $\bar{\bar{\Xi}}(\bar{\bar{\eta}},\mathbf{B})$ is the transverse resistive electric field tensor which is described in Section 2.2.5. M_i is the ion mass. The energy density is

$$e = \frac{p}{\gamma - 1} + \rho \frac{\mathbf{v} \cdot \mathbf{v}}{2} + \frac{\mathbf{B} \cdot \mathbf{B}}{2} \tag{2}$$

where $\gamma = c_p/c_v$ is the ratio of the specific heats. The non-dimensional tensors are the stress tensor ($\bar{\bar{\tau}}$), the electrical resistivity ($\bar{\bar{\eta}}$), and the thermal conductivity ($\bar{\bar{k}}$), and $\bar{\bar{\mathrm{I}}}$ is the identity matrix. The non-dimensional numbers are defined as follows:

$$\begin{array}{lll} \text{Alfvén Number :} & Al & \equiv V_A/V \\ \text{Reynolds Number :} & Re & \equiv LV/\nu \\ \text{Magnetic Reynolds Number :} & Rm & \equiv \mu_o LV/\eta \\ \text{Péclet Number :} & Pe & \equiv LV/\kappa \end{array} \tag{3}$$

The characteristic variables are length (L), velocity (V), Alfvén speed ($V_A = B/\sqrt{\mu_o \rho}$), kinematic viscosity ($\nu$), electrical resistivity (η), and thermal diffusivity ($\kappa = k/\rho c_p$). μ_o is the permeability of free space ($4\pi \times 10^{-7}$).

For convenience, the MHD equation set [eqn(1)] is rewritten in the following compact form

$$\frac{\partial Q}{\partial t} + \nabla \cdot \bar{\bar{T}}_h = \nabla \cdot \bar{\bar{T}}_p, \tag{4}$$

where Q is the vector of conservative variables, $\bar{\bar{T}}_h$ is the tensor of hyperbolic fluxes, and $\bar{\bar{T}}_p$ is the tensor of parabolic fluxes. The forms of these vectors and tensors can be seen from eqn(1). The hyperbolic fluxes are associated with wave-like motion, and he parabolic fluxes are associated with diffusion-like motion.

2.2.2 Algorithm Overview

Because of the natural differences between hyperbolic and parabolic equations, the methods for solving them are very different. Since the MHD equations are of mixed type the hyperbolic and parabolic terms must be handled differently. The hyperbolic fluxes are differenced by applying an implicit, approximate Riemann algorithm that properly accounts for their wave-like behavior. The parabolic terms are discretized by applying explicit central differencing.

The design of the overall algorithm is primarily driven by the numerical techniques that must be used to discretize the hyperbolic terms. Therefore, we begin by considering the ideal MHD equations, which are obtained from eqn(4) by setting all the parabolic terms ($\bar{\bar{T}}_p$) to zero.

In one dimension they are

$$\frac{\partial Q}{\partial t}+\frac{\partial F}{\partial x}=\frac{\partial Q}{\partial t}+A\frac{\partial Q}{\partial x}=0, \tag{5}$$

where F is the flux vector in the x direction (i.e. $\bar{\bar{T}}_h=(F,G,H)$) and A is its Jacobian.

$$A=\frac{\partial F}{\partial Q} \tag{6}$$

Here, Q is the vector of conserved variables:

$$Q=(\rho,\rho v_x,\rho v_y,\rho v_z,B_y,B_z,e)^T\,. \tag{7}$$

This is a set of hyperbolic equations and thus A has a complete set of real eigenvalues given by

$$\lambda=(v_x,v_x\pm V_{fast},v_x\pm V_{slow},v_x\pm V_{Ax})^T\,, \tag{8}$$

where V_{fast} and V_{slow} are the fast and slow magnetosonic speeds, and V_{Ax} is the Alfven speed based on the x component of the magnetic field. These can be expressed as

$$V_{fast}^2=\frac{1}{2}\left[c_s^2+V_A^2+\sqrt{\left(c_s^2+V_A^2\right)^2-4c_s^2V_{Ax}^2}\right], \tag{9}$$

$$V_{slow}^2=\frac{1}{2}\left[c_s^2+V_A^2-\sqrt{\left(c_s^2+V_A^2\right)^2-4c_s^2V_{Ax}^2}\right], \tag{10}$$

$$V_{Ax}^2=\frac{B_x^2}{\mu_o\rho}. \tag{11}$$

Here, c_s is the ion sound speed, which for a perfect gas is

$$c_s^2=\frac{\gamma p}{\rho}. \tag{12}$$

Information propagates along characteristics which travel at wave speeds given by the eigenvalues. Most modern numerical techniques for solving hyperbolic equations are based upon the idea of splitting the fluxes into components due to left and right running waves. Then each part of the flux can be differenced in an upwind manner, which greatly reduces numerical oscillations and stabilizes the solutions.

It is well known that if a hyperbolic equation is solved with an explicit scheme, then the allowable time step to maintain numerical stability is given by the CFL (Courant-Friedrichs-Lewy) condition, which in the case of the 1D MHD equations is

$$\Delta t < \frac{\Delta x}{|v_x + V_{fast}|}. \tag{13}$$

For the high magnetic fields and low densities common in many plasma experiments, the fast magnetosonic speed is quite high, and thus the time step is prohibitively small. We are often interested in only modeling the physics that occurs slower than Alfvén time scales. For example, it can be shown that resistive tearing modes, which are important in studying fusion plasmas, evolve on a time scale that is given by[4]

$$\tau_{tearing} \propto \tau_A^{2/5} \tau_\eta^{3/5} = (Lu)^{3/5}\,\tau_A. \tag{14}$$

τ_A is the Alfvén time, τ_η is the resistive diffusion time, and Lu is the Lundquist number, which is given by

$$Lu = \frac{\tau_\eta}{\tau_A} = RmAl. \tag{15}$$

If Lu is 10^6, which is typical for laboratory plasmas in fusion applications, the resistive tearing time is approximately 4000 times larger than the Alfvén time. By treating the hyperbolic fluxes implicitly in time, the stability restriction on the time step is removed, and the solution can be advanced at the larger resistive tearing time step. This is our motivation for proposing an implicit scheme.

The starting point for deriving the algorithm is the two-dimensional ideal MHD equations in Cartesian form

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0. \tag{16}$$

We then discretize eqn(16) using first order Euler time differencing to get

$$\frac{\left(Q_{ij}^{n+1} - Q_{ij}^{n}\right)}{\Delta t} = -R_{ij}\left(Q^{n+1}\right) = -R_{ij}^{n+1} \tag{17}$$

where R is

$$R_{ij} = F_{i+1/2,j} - F_{i-1/2,j} + G_{i,j+1/2} - G_{i,j-1/2}. \tag{18}$$

Note that in this equation and all that follow the grid metric terms (cell areas and volumes) are omitted for clarity. We linearize R as follows:

$$R_{ij}^{n+1} \approx R_{ij}^{n} + \left(\frac{\partial R}{\partial Q}\right)_{ij}^{n} \left(Q_{ij}^{n+1} - Q_{ij}^{n}\right) \tag{19}$$

Substituting this expression back into eqn(17) and rearranging, we get

$$\left[\frac{\bar{\bar{\mathrm{I}}}}{\Delta t} + \left(\frac{\partial R}{\partial Q}\right)_{ij}^{n}\right] \Delta Q_{ij}^{n} = -R_{ij}^{n}. \tag{20}$$

Here ΔQ is defined as

$$\Delta Q^{n} \equiv Q_{ij}^{n+1} - Q_{ij}^{n}. \tag{21}$$

The left hand side of the eqn(20) is an implicit operator operating on ΔQ. It is a large banded block matrix. In three dimensions, it is an $(I_{max} \times J_{max} \times K_{max})$ by $(I_{max} \times J_{max} \times K_{max})$ matrix, where I_{max} is the number of cells in the x direction, etc. It is quite costly to invert a matrix of this size directly. We choose to invert it using an approximate factorization, which can be done more efficiently. When solved this way, eqn(20) is no longer time accurate. However, we can still achieve time accuracy with this type of scheme by adding the time derivative of Q as a source term to the right hand side of the equation. We then have

$$\left(\frac{\partial Q}{\partial \tau}\right)^{n+1} = -R_{ij}^{n+1} - S_{ij}^{n+1} \tag{22}$$

where

$$S_{ij}^{n+1} = \frac{1}{2\Delta t}\left(3Q_{ij}^{n+1} - 4Q_{ij}^{n} + Q_{ij}^{n-1}\right) \approx \frac{\partial Q}{\partial t}. \tag{23}$$

The τ in eqn(22) can be thought of as a pseudo time variable. At each physical time step, eqn(22) is solved iteratively in pseudo time until the left hand side vanishes. When the solution converges, our original equation

$$\frac{\partial Q}{\partial t} = -R \tag{24}$$

is solved. This technique is known as dual time-stepping.[5] Note that in eqn(23) a more accurate time derivative can be used at the expense of the additional memory needed to store the Q vectors from previous time steps.

One advantage of the strategy outlined above is that the implicit operator and the right hand side in eqn(20) are decoupled. The structure of the matrix no longer depends on the details of the discretization of the right hand side fluxes. In the following sections we will describe the relaxation scheme that is used to iteratively invert the implicit operator and the approximate Riemann solver that is used to form the right hand side fluxes.

2.2.3 LU-SGS Relaxation Scheme

We use the lower-upper symmetric-Gauss-Seidel (LU-SGS) method to iteratively invert the implicit operator.[6] To derive this method, we first consider the following first order accurate flux-vector splitting of R (at time level $n+1$) in eqn(17):

$$R_{ij} = F^+_{ij} - F^+_{i-1,j} + F^-_{i+1,j} - F^-_{ij} + G^+_{ij} - G^+_{i,j-1} + G^-_{i,j+1} - G^-_{ij} \tag{25}$$

where F^+ is the portion of the F flux vector corresponding to right-running waves, and F^- is the portion corresponding to left-running waves, and G^+ and G^- are similarly defined. This equation is linearized to obtain

$$\begin{aligned}\left\{\bar{\bar{\mathrm{I}}} + \Delta t \left(A^+_{ij} - A^+_{i-1,j} + A^-_{i+1,j} - A^-_{i,j} + B^+_{ij} - B^+_{i,j-1} + B^-_{i,j+1} - B^-_{i,j}\right)\right\} \\ \times \Delta Q^n_{ij} = -\Delta t R^n_{ij}\end{aligned} \tag{26}$$

where A^+ is the Jacobian of F^+, and so on. We approximate these Jacobians as

$$A^+ = \frac{1}{2}(A + \rho_A) \tag{27}$$

$$A^- = \frac{1}{2}(A - \rho_A) \tag{28}$$

where ρ_A is the maximum eigenvalue (spectral radius) of A. If we then iteratively solve this simplified implicit operator using a forward Gauss-Seidel sweep followed by a backward sweep, the resulting algorithm can be written as

$$\begin{aligned}&\left\{\bar{\bar{\mathrm{I}}} + \Delta t \left[(\rho_A + \rho_B)\bar{\bar{\mathrm{I}}} - A^+_{i-1,j} - B^+_{i,j-1}\right]\right\} \\ &\qquad \times \left\{\bar{\bar{\mathrm{I}}} + \Delta t \left[(\rho_A + \rho_B)\bar{\bar{\mathrm{I}}} + A^-_{i+1,j} + B^-_{i,j+1}\right]\right\} \\ &\times \Delta Q^n_{ij} = -\left[1 + \Delta t(\rho_A + \rho_B)\right]\Delta t R^n_{ij} \quad .\end{aligned} \tag{29}$$

The forward sweep is equivalent to inverting a lower block diagonal matrix [the first braced term in eqn(29)], and the backward sweep is equivalent to inverting an upper block diagonal matrix [second braced term in eqn(29)].

This structure leads naturally to several vectorization and parallelization strategies.

If we sweep through the computational domain along lines of constant $i+j$ (in 2-D), each term along these lines is independent of the others and depends only on data that has already been updated during the current sweep. This type of fine grain parallelization is ideal for vector computers. However, that degree of parallelism is not efficient for parallel computers because the extra communication time between processors exchanging data more than offsets the gain in computational efficiency. To optimize this algorithm for a parallel architecture, we need to break up the computational domain into blocks and send each block to a different processor. At the boundaries between the blocks, we reduce the data dependency between the blocks by using data from the previous time step along the block boundaries. This effectively reduces those points into a Jacobi iteration. However, the interior points are still solved with a Gauss-Seidel iteration. As long as the blocks are large enough that there are many more interior points than boundary points, then the overall convergence rate is approximately the same as Gauss-Seidel.

2.2.4 Approximate Riemann Solver

The fluxes on the right hand side of eqn(20) are discretized using a Roe-type approximate Riemann solver.[7] In this method the overall solution is built upon the solutions to the Riemann problem defined by the discontinuous jump in the solution between each pair of cells. The numerical flux for a first-orde accurate (in space) Roe-type solver is written in symmetric form as

$$F_{i+1/2} = \frac{1}{2}\left(F_{i+1} + F_i\right) - \frac{1}{2}\sum_k l_k \left(Q_{j+1} - Q_j\right)|\lambda_k|\, r_k \qquad (30)$$

where r_k is the k^{th} right eigenvector, λ_k is the absolute value of the k^{th} eigenvalue, and l_k is the k^{th} left eigenvector. The values at the cell interface $(i+1/2)$ are obtained by a simple average of the neighboring cells. These first order accurate upwind fluxes are used in the vicinity of sharp discontinuities in order to suppress oscillations in the solution. We achieve a globally second order accurate solution by using a flux limiter that modifies the first order flux so that it uses second order central differencing in smooth portions of the flow. We are using a minmod limiter.[8]

Once the eigenvalues and eigenvectors are obtained and properly normalized to avoid singularities, it is relatively straight-forward to apply this scheme to the one-dimensional ideal MHD equations.[9,10] Unlike for the Euler equations, the extension to more than one dimension is non-trivial. The reason is that in more than one dimension, the Q vector must include B_x in addition to the other magnetic field components. (For the one-dimensional case B_x is constant by virtue of $\nabla \cdot \mathbf{B} = 0$). Since the $\mathbf{j} \times \mathbf{B}$ force acts perpendicularly to the directions of $\mathbf{j}$ and $\mathbf{B}$, the F flux vector has a zero term corresponding to B_x. Thus, the Jacobian matrix of F is singular and has a zero eigenvalue. This means we no longer have a complete set of physically meaningful eigenvectors. Physically, we expect information to travel either at the fluid velocity or at the fluid velocity plus or minus the wave speeds. Simply dropping B_x from the equation set is not a viable option, because B_x needs to change in order to maintain $\nabla \cdot \mathbf{B} = 0$. Powell et al., recently solved this problem by modifying the Jacobian in such a way as to change the zero eigenvalue to v_x (keeping the others unchanged), and then adding in a source term that exactly cancelled out the terms introduced by the modified Jacobian.[11]

The source term is

$$S_{div} = -\begin{bmatrix} \rho \\ \mathbf{B} \\ \mathbf{v} \\ \mathbf{v} \cdot \mathbf{B} \end{bmatrix} \nabla \cdot \mathbf{B} \tag{31}$$

It is proportional to the divergence of $\mathbf{B}$ and thus very small.

2.2.5 Parabolic MHD Terms

To this point we have only considered the hyperbolic terms. When finite viscousity and resistivity are included, the parabolic terms of the MHD equations [right hand side of eqn(1)] become important. For reasonably large values of Re and Rm (easily in the range of interest for most applications), the parabolic terms can be differenced explicitly without constraining the allowable time step. In this work we difference the parabolic terms explicitly in time with central differences in space. They are added to the right hand side fluxes arising from the approximate Riemann solver.

During the past year, we have spent a concerted effort on the parabolic terms to achieve accurate and stable calculations. Originally we used the same flux centering scheme that is used in MACH3 where the fluxes are calculated at the cell vertices and a divergence law is applied around the cell center. See Figure 1(a). A detailed stability analysis demonstrates the

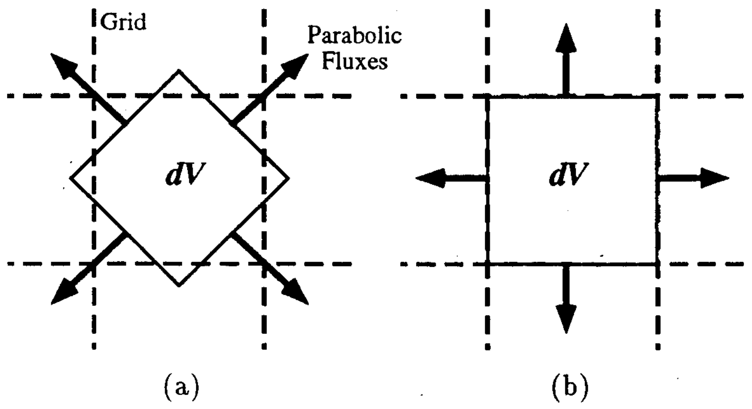


Figure 1: Positioning for (a) vertex-centered and (b) face-centered parabolic fluxes. The face-centered fluxes produced more accurate and stable solutions.

potential for grid decoupling and a resulting odd-even instability. [This result has important implications to all ALE (arbitrary Lagrangian-Eulerian) codes and will soon be submitted to a journal.] Locating the parabolic fluxes at the cell faces which corresponds to the location of the hyperbolic fluxes produced solutions that converged faster and were more accurate than locating the parabolic fluxes at the cell vertices. See Figure 1(b).

We also point out that the resistive electric field term in eqn(1) is different than the one commonly used and presented last year $\nabla \cdot \bar{\bar{\eta}} \cdot \nabla \mathbf{B}$ which does not hold for spatially dependent anisotropic resistivity. Plasma resistivity is a strong function of temperature and of the orientation to the magnetic field. Therefore, the assumption of spatially constant isotropic resistivity is incorrect. The new term reduces from the conservative formulation of the more general equation.

$$\int dV \nabla \times (\bar{\bar{\eta}} \cdot \nabla \times \mathbf{B}) = \oint dS \times (\bar{\bar{\eta}} \cdot \nabla \times \mathbf{B}) = \int dV \nabla \cdot \bar{\bar{\Xi}} = \oint dS \cdot \bar{\bar{\Xi}} \quad (32)$$

where the transverse resistive electric field tensor is defined as

$$\bar{\bar{\Xi}} \equiv \begin{bmatrix} 0 & \eta_z (\partial_x B_y - \partial_y B_x) & \eta_y (\partial_x B_z - \partial_z B_x) \\ \eta_z (\partial_y B_x - \partial_x B_y) & 0 & \eta_x (\partial_y B_z - \partial_z B_y) \\ \eta_y (\partial_z B_x - \partial_x B_z) & \eta_x (\partial_z B_y - \partial_y B_z) & 0 \end{bmatrix} \quad (33)$$

The dimensionless numbers have been removed for clarity.

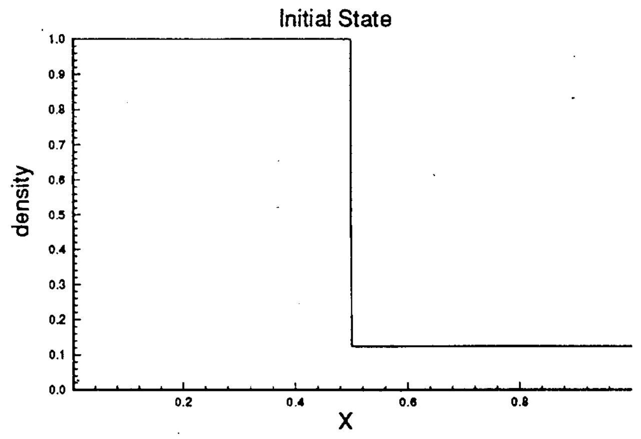
3 Benchmarks and Applications

3.1 Ideal MHD Test Problems

3.1.1 One-Dimensional Coplanar MHD Riemann Problem

This test problem served to validate the approximate Riemann solver, because the computed solution could be checked against the exact analytical solution. For the one-dimensional ideal MHD equations (variations in x only), the equation for B_x reduces to B_x is constant and drops from the equation set, eliminating the zero eigenvalue in this case. The coplanar MHD equations are obtained from the full one-dimensional ideal MHD equations by setting B_z and v_z to zero, thus allowing only planar flow and fields. This eliminates the $v_x \pm V_{Ax}$ eigenvalues, leaving a system of five equations with five eigenvalues. Mathematically, the Riemann problem is an initial boundary value problem in which there is initially a discontinuity in the data such that the left half of the domain is at one state and the right half of the domain is at another state. As the solution evolves in time, shock waves and rarefaction waves form and travel at speeds related to the wave speeds of the system. Although not physically realizable in plasmas, this problem is analogous to a shock tube in hydrodynamics.

For the full five-wave case, there is not a closed form analytical solution. Instead, the solution must be checked by calculating generalized Riemann invariants across the rarefaction waves and Rankine-Hugoniot jump conditions across the shock waves. Since this has already been done by Brio and Wu[9] for a specific set of conditions, for our test case we used the same initial conditions as they used in order to allow direct comparison with their published solution. The initial left state was $p = 1$, $\rho = 1$, and $B_y = 1$. The initial right state was $p = 0.1$, $\rho = 0.125$, and $B_y = -1$. The velocities were zero and B_x was 0.75. Figure 2 shows the initial density distribution and its numerical solution after 400 time steps on an 800 point grid with a CFL number of 0.8. Figure 3 is the corresponding plot of the transverse magnetic field (B_y). The solution was computed using explicit time-stepping. The solution clearly shows five waves formed corresponding to the five eigenvalues. They are a fast rarefaction wave, a slow shock, a contact surface moving to the right, a slow compound wave (rarefaction and shock), and a fast rarefaction wave moving to the left. Note that the numerical method is able to resolve the shocks over a few grid points without introducing numerical oscillations. This is one of the advantages of the flux splitting approach we have used. The computed solution overlaid exactly on Brio and Wu's



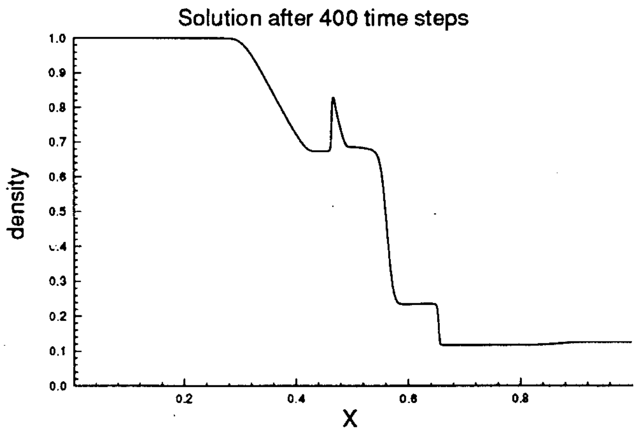


Figure 2: Numerical solution of coplanar Riemann problem. Density profile is shown initially and after solution has evolved for 400 time steps.

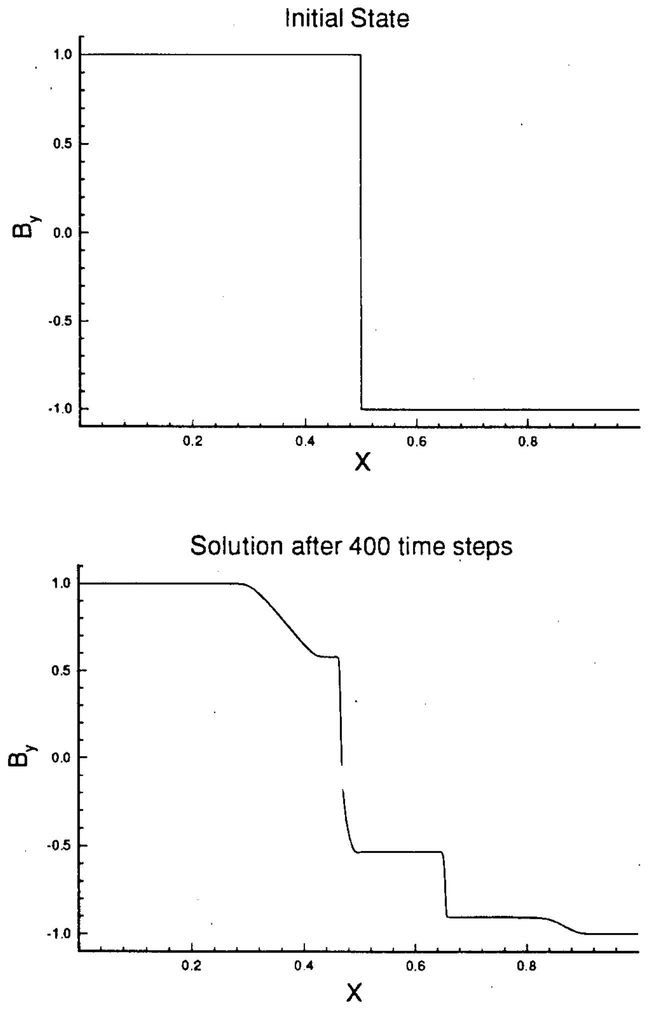


Figure 3: Numerical solution of coplanar Riemann problem. Transverse magnetic field profile is shown initially and after solution has evolved for 400 time steps.

published solution.

If we set $B_x = 0$ above, then the problem reduces to a hydrodynamic shock tube problem if one replaces the thermodynamic pressure by the sum of the thermodynamic and magnetic pressures. For this case one can find a closed form exact solution to compare to the calculated solution. Figure 4 shows both the calculated and the exact solution for $p + B^2/2$ after 80 time steps on a 100 point grid. There is very good agreement with the plateau values and the shock is resolved in a few cells without numerical oscillations.

3.1.2 Oblique Shock

This steady-state problem served primarily as a test of the LU-SGS implicit relaxation scheme. It also allowed us to examine the divergence of **B** at each point to ensure that the the zero eigenvalue fix was correctly implemented.

The geometry for these tests is shown in Figure 5. A super-Alfvénic flow (Mach number of 3) impinges on a perfectly conducting plate at an angle of 25 degrees. In addition, a vertical field of $B_y = 0.2$ is imposed at the left boundary. Since the plate is perfectly conducting, the component of the magnetic field normal to the plate is held at zero.

Figure 6 shows the steady-state solution of this problem. Contours of density and magnetic field lines are plotted. The density contours show that an oblique shock forms, as expected. Outside of the shock, the field is convected in from the boundary. At the shock, the field lines bend due to the change in direction of the flow at the shock. Finally, the field lines bend at the conducting wall as all the field is converted to B_x to satisfy the boundary condit'on while keeping the divergence of **B** equal to zero. We verified that the divergence was less than 10^{-14} throughout the domain.

This two-dimensional steady-state solution was obtained with explicit time stepping at a CFL number of 0.8 and with the LU-SGS implicit relaxation scheme at an infinite CFL number (approximate Newton iteration). Figure 7 is a plot of the logarithm of the two-norm of the residual of the energy equation as a function of the number of iterations. The implicit scheme converged to 10^{-14} in about 150 iterations, whereas the explicit scheme required about 700 iterations. This is an acceleration factor of about 4.5 for the implicit scheme. Higher acceleration factors can be achieved for finer grids.

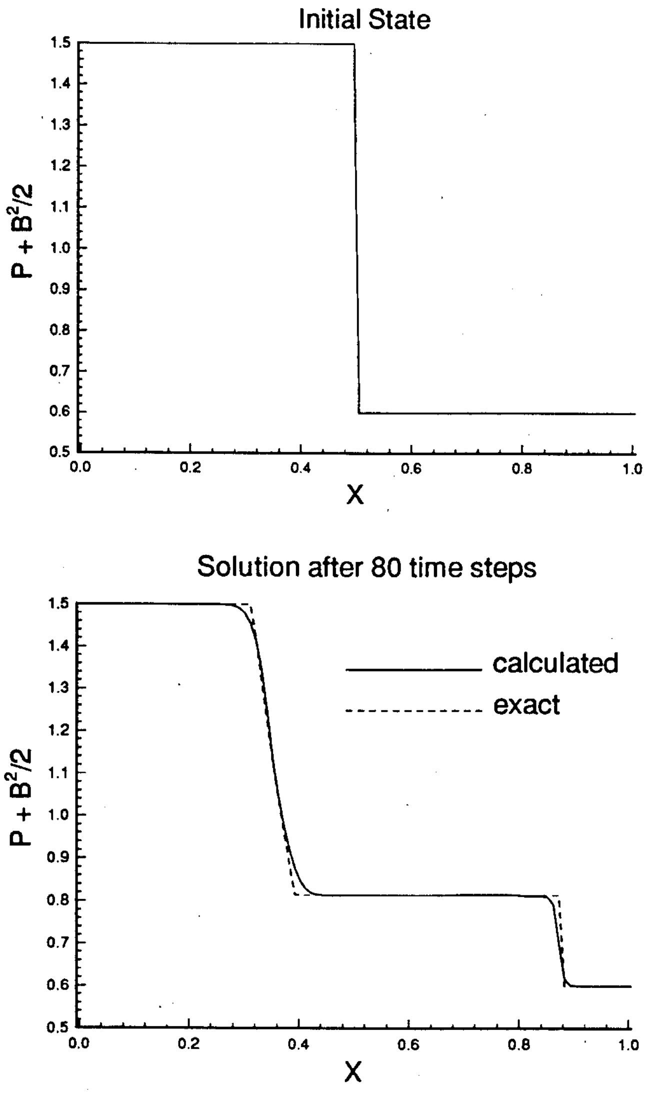


Figure 4: Comparison of numerical and exact solution of coplanar Riemann problem for $B_x = 0$ case.

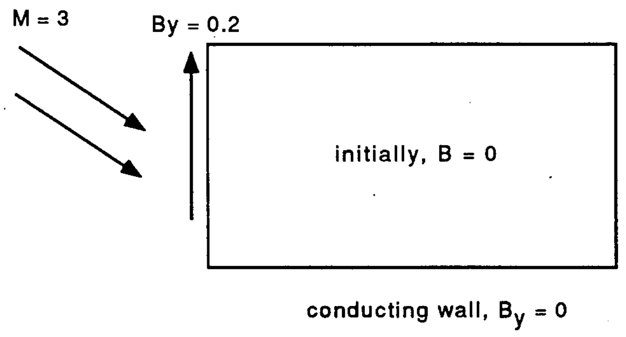


Figure 5: Geometry of oblique shock test problem.

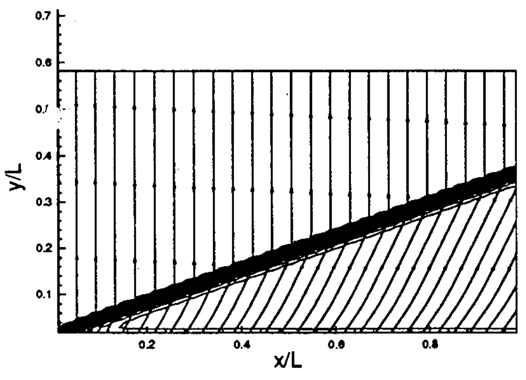


Figure 6: Density contours and field lines for an $M = 3$ flow impinging on a perfectly conducting plate at an angle of 25 degrees.

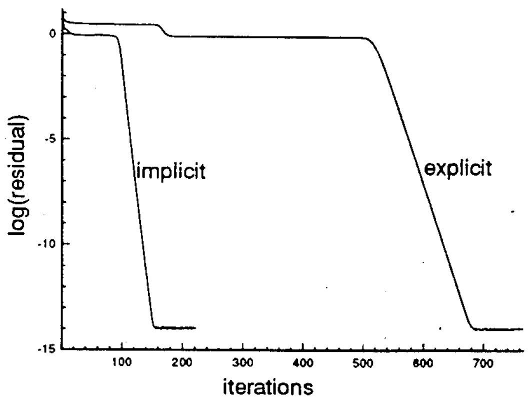


Figure 7: Logarithm of the two-norm of the energy equation residual plotted as a function of iteration number for explicit and implicit solutions of channel flow with horizontal velocity and vertical magnetic field imposed at the left boundary.

3.2 Viscous and Resistive MHD Test Problems

The viscous and resistive terms in the MHD equations comprise the right hand side of the equality in eqn(1). The addition of these terms to the algorithm involved the modification of the R vector in eqn(20).

$$-R \rightarrow -R + \nabla \cdot \bar{\bar{T}}_p \tag{34}$$

The R vector is updated with each iteration to produce a solution that is fully coupled.

Using the divergence form of the parabolic terms reduces the differencing errors of the method. To preserve the accuracy on irregular meshes the derivatives are computed using a finite volume method.

The validation of the parabolic terms consisted of applying the code to a suite of test problems with known analytical solutions. We validated independently the terms associated with viscosity and those associated with resistivity and then the combined effect of all of the terms. The test problems were: (1) fully developed laminar flow between two parallel plates, (2) magnetic field generated by a constant current density, and (3) Hartmann flow. All of these test problems were run until a steady-state solution developed. The capability of the code to capture time-dependent physical effects was also tested by modeling the exponential resistive decay of the magnetic field generated in test problem 2.

3.2.1 Laminar Flow

We benchmarked the code to two types of laminar flows between infinite parallel plates. The plates restrict the steady-state flow to be one-dimensional. No magnetic fields are present. This reduces the MHD equations to the Navier-Stokes equations. In these simulations a no-slip boundary condition was applied to the fluid in contact with the plates.

The first type of flow to which we benchmarked was viscous flow generated by one plate moving relative to the other plate. With no pressure gradient, constant viscosity, and incompressible flow, the equations reduce to

$$(Re)^{-1}\,\nabla\cdot\bar{\bar{\tau}} = (Re)^{-1}\,\nabla^2 v_x = 0 \tag{35}$$

which is Laplace's equation. For finite viscosity (Re) the analytical solution for the flow velocity is

$$v_x(y) = V_0\left(1 - \frac{y}{L}\right) + V_L\frac{y}{L} \tag{36}$$

where V_0 is the velocity of the plate at $y = 0$ and V_L is the velocity of the plate at $y = L$.

The errors between the analytical solution and the code generated solution were below 10^{-9} (the two-norm of the -rror between the solutions). We performed the same simulation with no viscosity ($Re \to \infty$). As would be expected, the flow velocity vanished everywhere except on the plates. When the viscous heating was modeled, a transient pressure gradient $p(y)$ and transverse velocity $v_y(y)$ developed which heated the flow and increased its energy.

The next test was laminar flow between stationary parallel plates with a constant pressure gradient in the flow direction. The governing equation is

$$\nabla_x\cdot\left(p\bar{\bar{\mathrm{I}}}\right) = \frac{\partial p}{\partial x} = (Re)^{-1}\,\nabla^2 v_x. \tag{37}$$

The solution for this flow is the parabolic equation given by

$$v_x(y) = (Re)\,\frac{\partial p}{\partial x}\left(\frac{y}{L}\right)\left(\frac{L-y}{2L}\right). \tag{38}$$

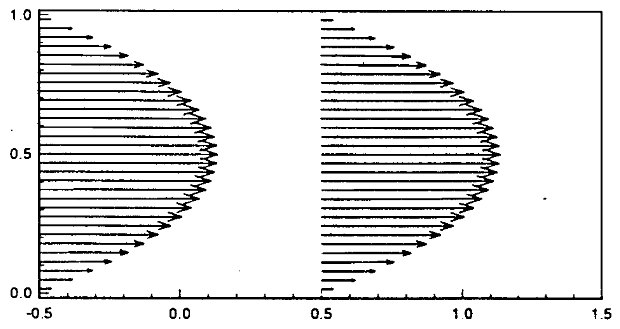


Figure 8: Simulation of laminar flow between parallel plates in the presense of a constant pressure gradient. The velocity profile is parabolic as expected from the analytical solution.

Figure 8 shows the solution generated by the code. Again the errors were reduced to below 10^{-9}.

3.2.2 Resistive Diffusion

We benchmarked the resistive diffusion to a current sheet with a uniform current density. Values of the tangential magnetic field were specified at parallel infinite plates, in a similar way as the first of the laminar flow simulations.

For no flow velocity and constant resistivity the MHD equations reduce to a Laplace equation similar to eqn(35).

$$(Rm)^{-1}\,\nabla\cdot\nabla\mathbf{B}=0 \tag{39}$$

This equation has the same form for its solution as eqn(36).

$$B_x(y)=B_0\left(1-\frac{y}{L}\right)+B_L\frac{y}{L} \tag{40}$$

where B_0 is the velocity of the plate at $y=0$ and B_L is the velocity of the plate at $y=L$.

The code agreed with the analytical solution to within errors of 10^{-9}.

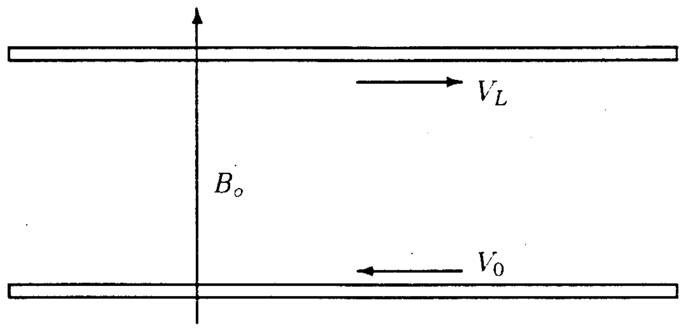


Figure 9: The Hartmann flow geometry showing the moving parallel plates and the cross magnetic field.

The time-dependent resistive decay of a magnetic field can be represented analytically by solving the one-dimensional transverse magnetic induction equation with constant resistivity.

$$\frac{\partial B_\perp}{\partial t} = (Rm)^{-1} \frac{\partial^2 B_\perp}{\partial x^2} \tag{41}$$

The solution is the exponential decay of the magnetic field with a sinusoidal profile.

$$B_\perp(t, x) \propto \exp\left(-\frac{\pi^2 t}{Rm}\right) \sin(\pi x) \tag{42}$$

for zero field boundary conditions at $x = 0$ and $x = 1$.

This simulation was performed beginning with a uniform field profile. The field decayed into the expected sinusoidal shape and the decay constant agreed with the analytical result to within 0.01%. The same test was repeated on a parabolic clustered grid with $\Delta x_{max}/\Delta x_{min} = 10$. The same accuracy was achieved.

3.2.3 Hartmann Flow

Hartmann flow combines the effects of viscosity and resistivity. The problem geometry is the same as that for the laminar flow with the addition of a magnetic field that is normal to the plates, in the y direction. See Figure 9 for an illustration.

The governing equations for the Hartmann flow can be found by combining the magnetic field and momentum equations from the MHD model.

As before there will only be flow in the x direction. However, an applied electric field in the z direction must be included since it can generate an $\mathbf{E_o} \times \mathbf{B_o}$ flow in the x direction. The Hartmann flow is described by the differential equation,

$$\frac{\partial^2 v_x}{\partial y^2} - \frac{H^2}{L^2}\left(v_x + \frac{E_o}{B_o}\right) = 0, \tag{43}$$

where the Hartmann number is defined as

$$H \equiv \frac{B_o L}{\sqrt{\rho \nu \eta}} = AlL\sqrt{ReRm}. \tag{44}$$

The analytical solution to the Hartmann flow is

$$\begin{aligned} v_x(y) = {} & V_0 \frac{\sinh\left(H(1-y/L)\right)}{\sinh(H)} + V_L \frac{\sinh(Hy/L)}{\sinh(H)} \\ & - \frac{E_o}{B_o}\left[1 - \frac{\sinh\left(H(1-y/L)\right) + \sinh(Hy/L)}{\sinh(H)}\right] \end{aligned} \tag{45}$$

where the same no-slip boundary conditions have been applied. In the limit of no magnetic field, the solution reduces to the laminar flow solution, eqn(36).

The response of the magnetic field can be determined by solving the magnetic field equation for the field component that will be "dragged" with the flow. This magnetic field is described by

$$\frac{\partial B_x}{\partial y} = -(Rm)\left(v_x + \frac{E_o}{B_o}\right). \tag{46}$$

Using the flow solution of eqn(45), the solution for B_x is

$$B_x(y) = \left(\frac{Rm}{H}\right)\left(\frac{V_L - V_0}{2}\right)\left[\frac{\cosh(H/2) - \cosh\left(H(L-2y)/2L\right)}{\sinh(H/2)}\right]. \tag{47}$$

The boundary conditions are that B_x vanishes at the plates and the net current is zero. The first boundary condition may seem arbitrary, but it is consistent with the no-slip boundary condition applied to the flow solution. The second boundary condition relates the applied electric field, E_o, and the plate velocities.

$$\frac{E_o}{B_o} = -\frac{V_0 + V_L}{2} \tag{48}$$

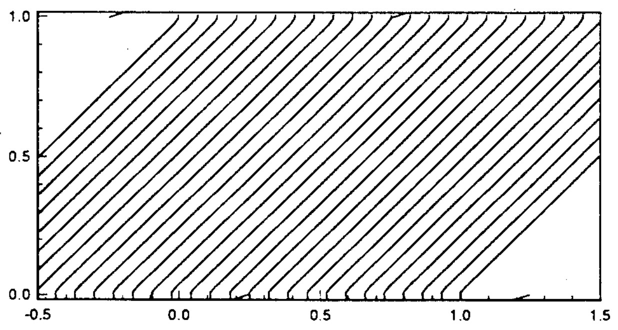


Figure 10: Hartmann flow simulation with $H = 10^4$. Flow velocity vectors and magnetic field lines are shown. The velocity of the flow is zero everywhere except at the plates. The magnetic field lines have a constant slope through the domain.

Since the MHD equation set does not allow for an applied electric field, V_0 is set to $-V_L$, so that $E_o = 0$.

We performed simulations for large, small, and intermediate Hartmann numbers, H.

For a large Hartmann number, the effects of viscosity and resistivity are small, and the solution approaches that of idea' MHD. The flow velocity vanishes everywhere except on the plates, like it does for the inviscid case ($Re \rightarrow \infty$). The magnetic field is frozen into the plates and develops a slope (constant B_x) as the plates move. The slope of the magnetic field is determined by the value of H (the field lines slip through the plates due to resistivity). The slope of the magnetic field lines (B_o/B_x) is constant at H/Rm. Figure 10 shows the results from simulation with $H = 10^4$. A finite value of the flow velocity exists only at the plates. The magnetic field lines are straight except at the plates where B_x is forced to vanish because of the boundary conditions. For clarity the slope of the magnetic field has been normalized to unity at the midplane between the plates for all of the Hartmann flow simulations.

The limiting case of small Hartmann number is characterized by a flow that is dominated by viscous effects and a magnetic field that responds to

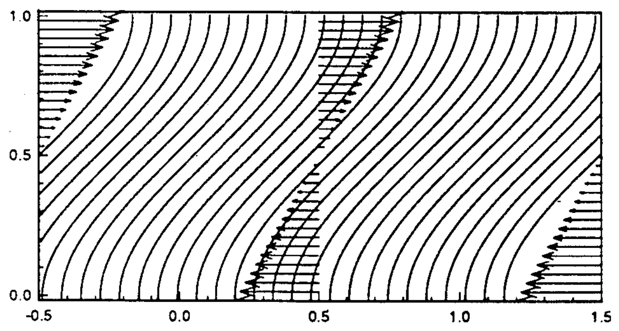


Figure 11: Hartmann flow simulation with $H = 0.1$. Flow velocity vectors and magnetic field lines are shown. The velocity profile is linear and the magnetic field lines have an "S" shape caused by the bulk fluid flow.

the bulk fluid flow and the large resistivity. The flow velocity varies linearly from the velocity of the top plate to the velocity of the bottom plate, as described by eqn(36). The magnetic field diffuses through the plate and the bulk fluid, but the fluid drags the field lines along with the flow. This produces a swayed "S" shape to the field lines with a peak magnetic field at the midplane. Since the slope of the field lines is inversely proportional to the magnitude of B_x, the peak in the magnetic field corresponds to the field lines with the minimum slope (most horizontal). The minimum slope is $4/Rm$. The simulation results for $H = 0.1$ are shown in Figure 11. Notice the linear velocity profile and the swayed magnetic field lines.

Flows with Hartmann numbers in the intermediate ranges have solutions which exhibit characteristics of both of the limiting cases. The flow velocity falls to zero away from the boundaries in a scale length of L/H. This scale length is an appreciable fraction of the domain. The magnetic field is influenced by the motion of the plates and the fluid flow. The magnetic field has a swayed shape close to the plates and is linear around the midplane. Away from the boundaries ($L/H < y < L-L/H$), the value of B_x is constant at $B_o Rm/H$. Figure 12 shows the results from a simulation with $H = 10$. The velocity profile falls to zero around the midplane. The magnetic field lines have a swayed shape like those in Figure 11 but not as dramatic, and

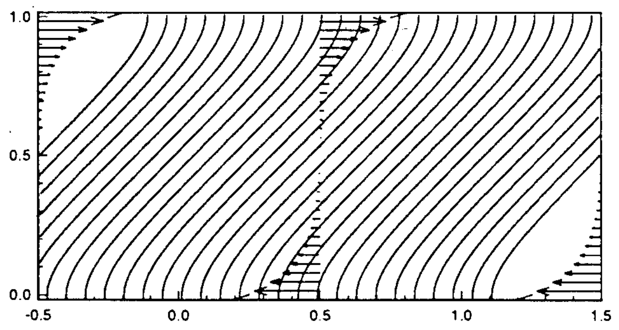


Figure 12: Hartmann flow simulation with $H = 10$. Flow velocity vectors and magnetic field lines are shown. The flow velocity only exists close to the plates. The magnetic field lines are linear around the midplane.

they are linear around the midplane.

All of the Hartmann flow simulations converged to the analytical solution to within errors of 10^{-6}.

3.3 MPD Plasma Thruster

The magnetoplasmadynamic (MPD) thruster is an electric propulsion device for spacecraft. Electrical propulsion is a technological field that is important to the Air Force and industry for satellite station keeping and orbital maneuvering. This problem demonstrates the dual time-stepping algorithm, which allows flexible choice of time steps so that fast and slow transients can be tracked accurately and efficiently. This is also the first problem that exercises all of the parts of the new algorithm (the approximate Riemann solver, the LU-SGS relaxation scheme, the resistive and viscous terms, and the dual time-stepping) simultaneously. The problem geometry is shown in Figure 13. A current is applied across the left boundary. This current creates a magnetic field in the z direction that in turn leads to a $\mathbf{j} \times \mathbf{B}$ force that accelerates the plasma to the right. We expect that the plasma initially in the domain will be accelerated up to some exit velocity on a fast time scale related to the Alfvén time. However, if there is a finite resistivity in

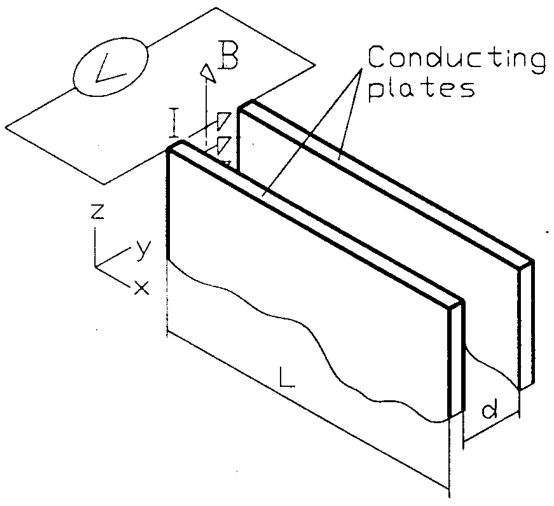
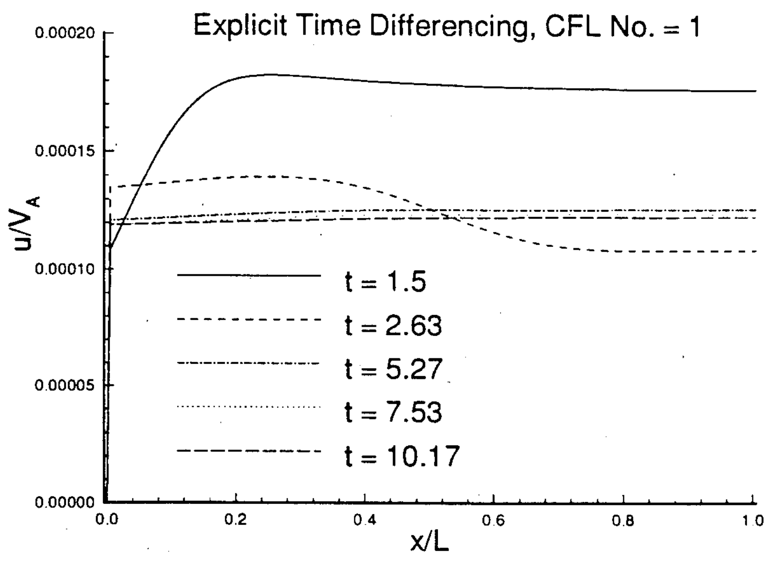


Figure 13: Geometry of the two-dimensional MPD thruster.

the plasma, the magnetic field and current at the left boundary will diffuse into the domain on a slower time scale related to the resistive diffusion time. Ideally, one would like to take small time steps initially to follow the fast transient, and then switch to a much larger time step when the system is evolving more slowly.

If there is no viscosity, then . .e problem becomes one-dimensional in x, which is to the right in Figure 13. For this problem we chose a Lundquist number of 100, a reference magnetic field of 1 Tesla, a reference density of 10^{-5} kg/m^3, a reference length of 10 cm, and an imposed current of 30 kA. Figure 14 shows the plasma velocity as a function of x at several different times (normalized to the Alfvén time). The top plot shows the results of an explicit time-differencing simulation with a CFL number of 1. This simulation took 2600 time steps to advance the solution to t = 10.17. Notice that between 3 and 5 Alfvén times, the velocity reaches a constant uniform value along the length of the domain. The bottom plot is a simulation in which a CFL number of 1 was used until t = 1.5, at which point the CFL number was increased to 100 and the dual time-stepping implicit method was used to maintain stability. At each physical time step



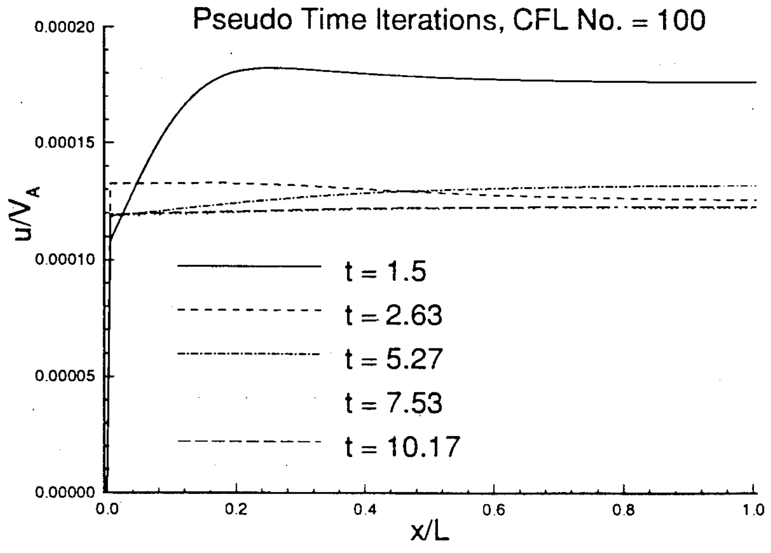
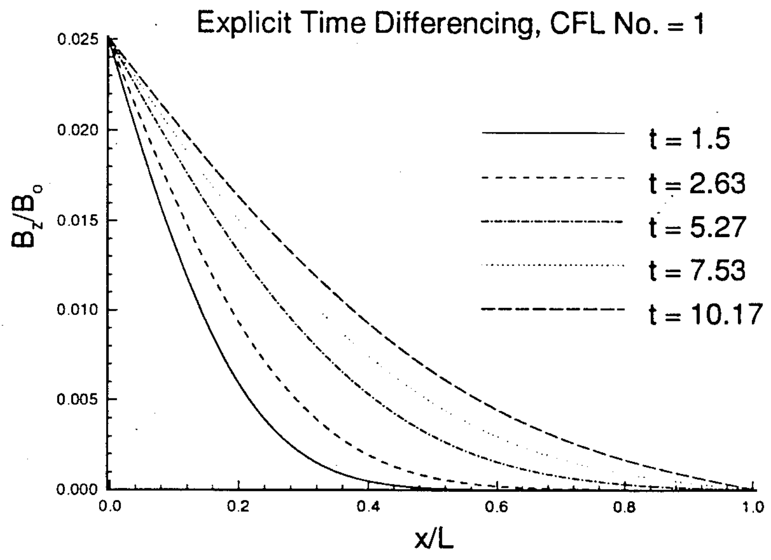


Figure 14: Plasma velocity as a function of x and time for explicit time differencing simulation and dual time-stepping (implicit) simulation.



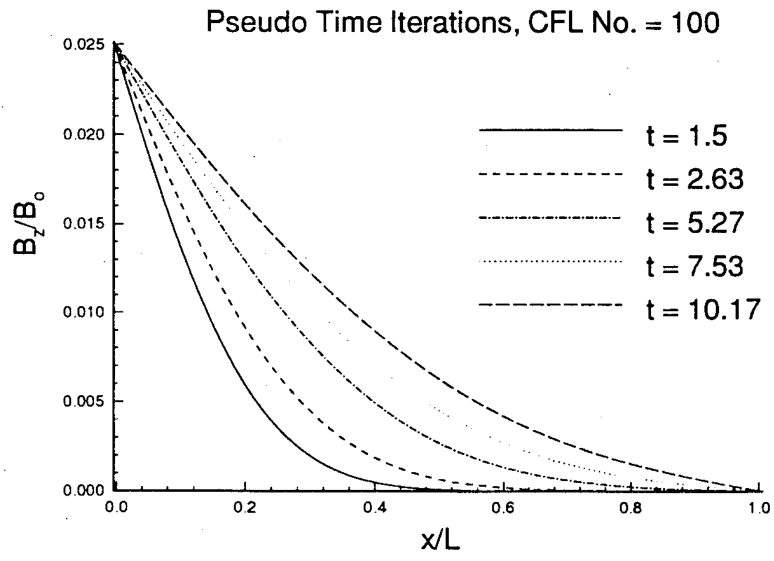


Figure 15: Magnetic field as a function of x and time for explicit time differencing simulation and dual time-stepping (implicit) simulation.

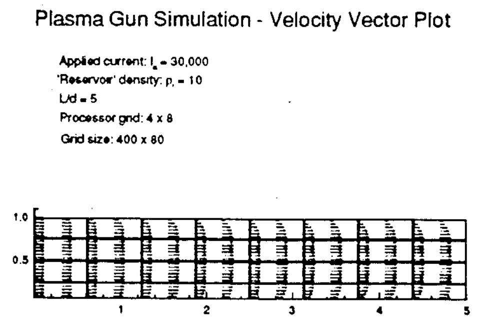


Figure 16: Velocity vectors for the MPD thruster. The plasma is accelerated down the gun by the $\mathbf{I} \times \mathbf{B}$ force and a boundary layer develops. The internal blocks illustrate the decomposition of the domain used for the validation of the parallel version of the code.

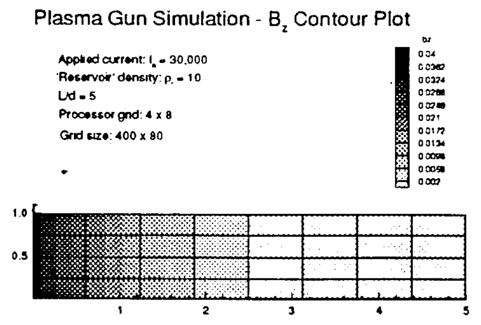


Figure 17: Magnetic field (B_z) contours for the MPD thruster. The gradient in the magnetic field produces the force applied to the plasma.

it took about 30 pseudo-time steps to converge, so the overall number of iterations was reduced to 1090 for the dual time-stepping case. The plots look similar to the explicit time-stepping results, except that the end of the fast transient is filtered out by taking such large time steps. On the other hand, Figure 15 shows that the magnetic field, which evolves on the slower resistive diffusion time scale, is captured equally well by the explicit and implicit schemes. The development of the plasma velocity and internal magnetic field can be seen in Figures 16 and 17.

3.4 Magnetic Reconnection

In this application we present results demonstrating agreement between theoretical linear growth rates of the resistive instability in a sheet pinch and our non-linear resistive MHD code. We study resistive instabilities because they are a likely candidate for driving magnetic relaxation in the Helicity Injected Tokamak (HIT). The planar sheet pinch is a well understood configuration[12–15] and provides a good test problem and benchmark for our MHD code.

We present the linear analysis of the sheet pinch.[13,14] The linear equations are solved numerically to obtain the eigenmodes. The eigenvalues (growth rates) are compared with the analytical theory.[12] We then present the nonlinear analysis where our implicit MHD code is applied. A perturbation is initialized in the MHD code. The instability resulting from the perturbation is allowed to develop and finally saturates due to non-linear effects. The initially linear growth rate agrees with linear analysis.

3.4.1 Problem Description

We study the resistive instability in a planar sheet pinch, the symmetric tearing mode in a finite-thickness current sheet. See Figure 18 for schematic representation. For simplicity we examine the mode with the wave vector parallel to the equilibrium magnetic field.

$$k \parallel B_o \tag{49}$$

We define

$$F \equiv \frac{\mathbf{k}\cdot\mathbf{B}}{\mathbf{B_{ref}}} = \tanh\left(\frac{y}{a}\right) \tag{50}$$

where a is the characteristic width of the current sheet. The resistivity of

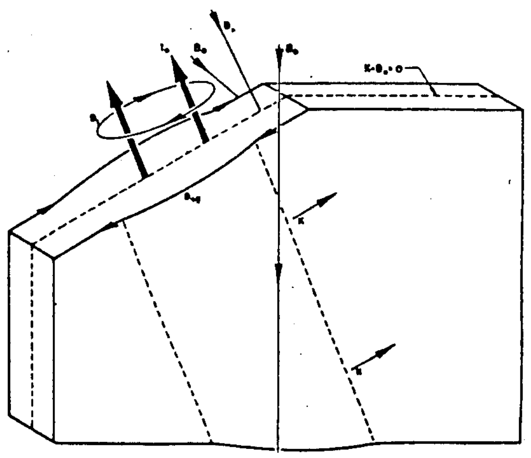


Figure 18: Schematic of planar sheet pinch problem [from H. P. Furth, Phys. Fluids **28**(6), 1595 (1985)].

the current sheet is

$$\frac{\eta}{\eta_{ref}} = \cosh^2\left(\frac{y}{a}\right) \tag{51}$$

which satisfies the equilibrium induction equation with no flow. The resistivity has a minimum in the middle of the current sheet ($y = 0$), and the magnetic field vanishes at $y = 0$ and is positive for $y > 0$ and negative for $y < 0$. See Figure 19 for the equilibrium profiles.

3.4.2 Linear Analysis

For the linear analysis, we begin with the incompressible, resistive MHD equations. We assume a variation of the perturbations of the form

$$f = f(y,t)e^{ikx}. \tag{52}$$

The perturbation equations yield a pair of coupled, linear differential equations.[14]

$$\frac{\partial \Psi}{\partial t} = \eta\left(\frac{\partial^2 \Psi}{\partial y^2} - \alpha^2 \Psi\right) - Fw \tag{53}$$

$$\frac{\partial}{\partial t}\left(\frac{\partial^2 w}{\partial y^2} - \alpha^2 w\right) = \alpha^2 Lu^2 \left[F\left(\frac{\partial^2 \Psi}{\partial y^2} - \alpha^2 \Psi\right) - \frac{\partial^2 F}{\partial y^2}\Psi\right] \tag{54}$$

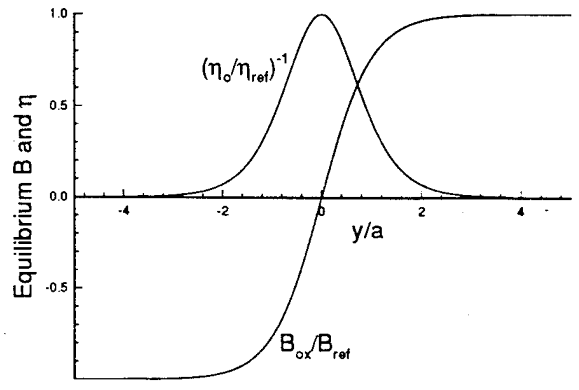


Figure 19: Equilibrium profiles of normalized magnetic field and resistivity.

where Lu is the Lundquist number and

$$\Psi \equiv \frac{B_{y1}}{B_o} \tag{55}$$

$$w \equiv -ik\tau_r v_{x1} \tag{56}$$

$$\alpha = ka \tag{57}$$

$$\tau_r = Lu\tau_A \tag{58}$$

This coupled pair of PDE's are solved numerically using an implicit finite difference formulation. The eigenfunctions for $Lu = 10^3$ and $\alpha = 0.5$ are shown in Figure 20. The growth rates have also been found analytically.[12] For the pure symmetric tearing mode the growth rate is given by

$$\gamma = 0.954(1-\alpha^2)^{4/5}\left(\frac{Lu}{\alpha}\right)^{2/5}. \tag{59}$$

For values of Lu greater than 500, the numerically calculated and analytical linear growth rates agree to within a few percent.

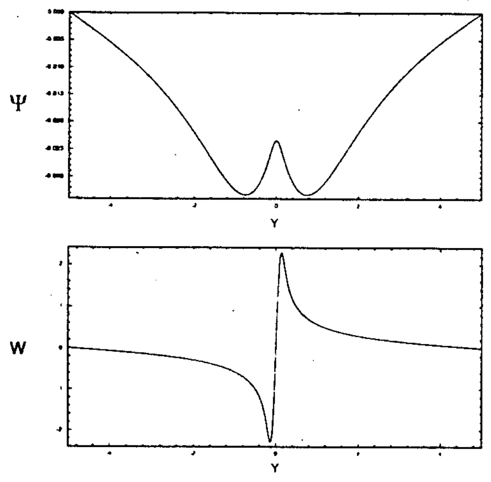


Figure 20: The eigenfunctions for $Lu = 10^3$ and $\alpha = 0.5$.

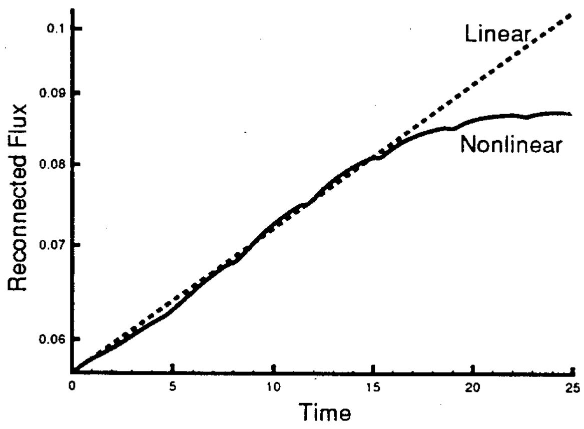


Figure 21: The linear and non-linear evolution of the reconnected flux.

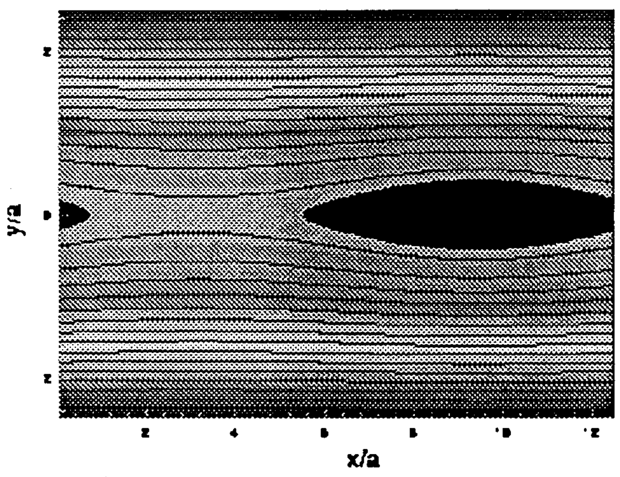


Figure 22: Flux contours of the developed non-linear instability.

3.4.3 Non-Linear Analysis

A planar current sheet is initialized in the code, and a perturbation of the same form as the linear eigenfunction is superimposed. The growth rate is determined from the amount of reconnected flux at $y = 0$. The evolution of the non-linear perturbation is shown in Figure 21. The result from the linear analysis is also shown. The non-linear growth matches the linear prediction during early development of the instability, but during late time the instability saturates due to non-linear effects. Magnetic flux contours are shown in Figure 22 which show the magnetic island formation of the non-linear instability.

3.5 HIT Injector

One of the first applications for the new code will be to simulate the HIT experiment. The experiment is shown schematically in Figure 23. The geometry is toroidal, but only a single slice in the poloidal plane is pictured. HIT is a low aspect ratio tokamak that uses helicity injection to produce toroidal current. Gas is puffed into the injector and then a series of capacitor banks are discharged across the electrodes to form the plasma and interact with the applied magnetic fields to push the plasma into the confinement

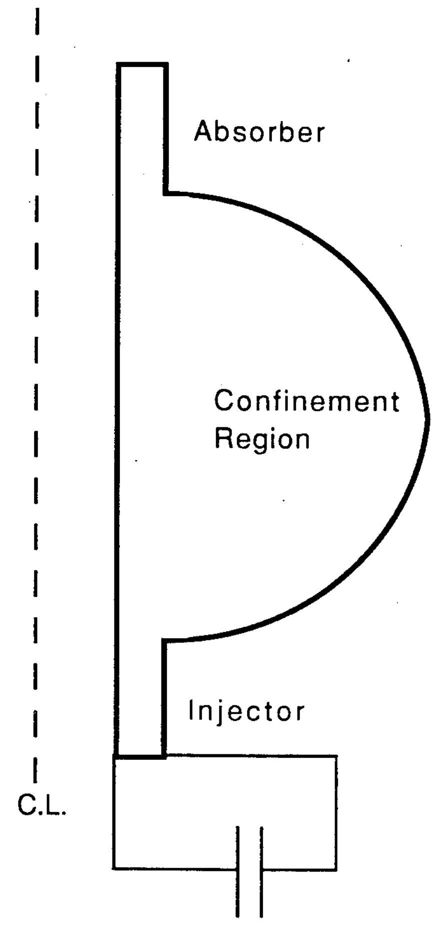


Figure 23: Schematic of the HIT plasma experiment.

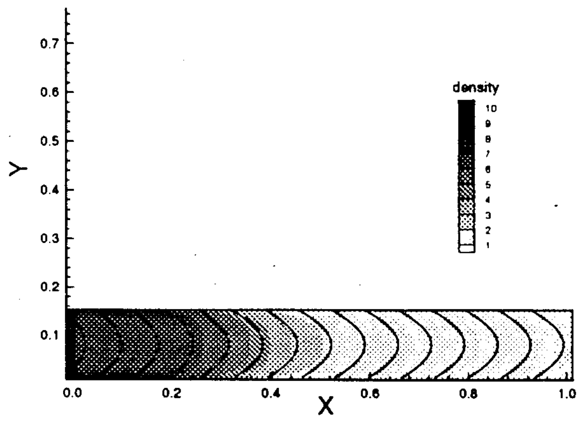


Figure 24: Results of two-dimensional simulation of HIT injector. Plot shows density contours and poloidal magnetic field lines.

region, where the tokamak plasma is formed. The full simulation will require three-dimensional, multiblock capability, which the code does not yet have. However, the injector portion of the experiment can be modeled as a single block.

For the first simulation we made the further simplification of solving the two-dimensional problem rather than the true cylindrical problem. We chose an initial poloidal bias field that is uniform throughout the injector with $B_y = 0.001T$ and $B_x = 0$, where the x direction is up (toward the confinement region) in Figure 23. The initial toroidal (out of plane) field was zero in this case, and a current of $30kA$ was applied across the electrodes (at the bottom boundary in Figure 23). Plasma was placed at the bottom boundary with a density ten times higher than the initial background density. The Lundquist number was 1000. A grid with 44 cells in the x direction and 12 points in the y direction was used.

The results of the simulation after ten Alfvén times are shown in Figure 24. Contours of density and the in-plane magnetic field lines are plotted. The current at the left boundary ($x = 0$) induces out-of-plane magnetic field that results in a $\mathbf{j} \times \mathbf{B}$ force that brings in plasma from the left and pushes the plasma to the right (in the x direction) towards the confinement region.

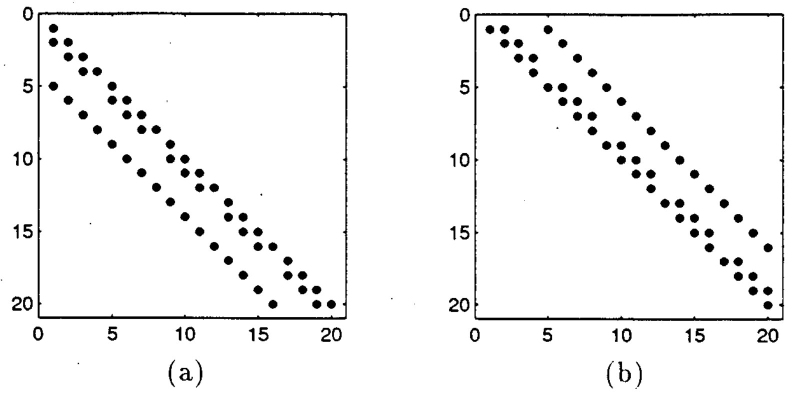


Figure 25: The 20×20 lower (a) and upper (b) tridiagonal block matrices for the LU-SGS algorithm with a grid of 4×5 cells.

The contours of density show the higher density plasma being carried into the domain. In addition, the initially straight field lines are stretched as the plasma flows across them. However, the density contours do not overlay with the field lines as they would in the limit of zero resistivity. This is consistent with the relatively low Lundquist number for this simulation.

4 Parallel Computer Implementation

We have begun to investigate strategies for implementing the algorithm on parallel architectures. The first of the following sections describes our first and the simplest approach, which was to parallelize the LU-SGS algorithm in a point-wise manner. This proved to be too fine-grained to be efficient, so we have since opted for a domain decomposition approach which is described in the second section. The third section describes the implementation of this method on the MHD solver.

4.1 Fine-Grain Parallelization

The LU-SGS algorithm involves a double sweep of the computational domain. The forward (predictor) sweep solves a lower tridiagonal block matrix for the entire computational domain. The backward (corrector) sweep solves an upper tridiagonal block matrix. Figure 25 shows the form of the lower and upper block diagonal matrices for the case of a 4×5 grid. Because

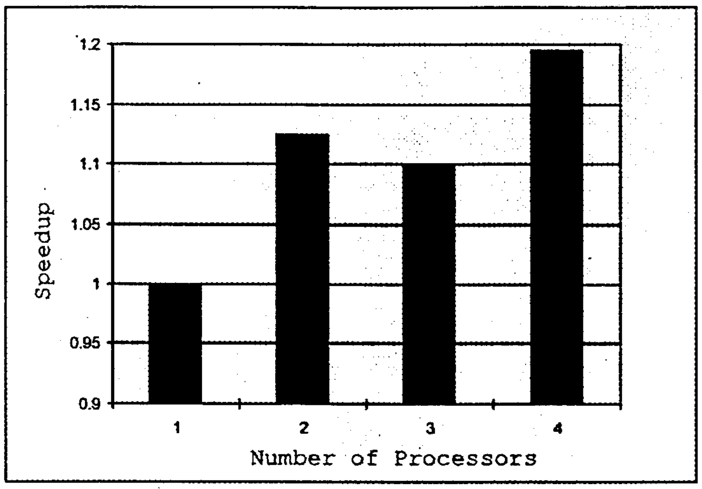


Figure 26: The parallel speedup for a problem with constant size grid using a fine-grain parallelization approach.

of the lower-upper form of the matrices, the solutions at grid cells along a line of constant $i + j$ are independent.

The simplest parallel implementation is to decompose the domain into its component cells, distribute the grid cells over the processors of the parallel computer, and treat each cell as residing on a different processor. This approach exploits the independence of the solutions of the cells on lines of constant $i+j$. Communication between the cells provides the necessary synchronization. For these tests, we used the Parallel Virtual Machine (PVM) communication library which was developed at Oak Ridge National Laboratory.[16] PVM allowed us to connect a network of four DEC Alpha workstation and use them as our parallel computer.

To determine the parallel effectiveness, we measured the speedup obtained when a problem grid of constant size was evenly distributed onto an increasing number of processors. Speedup is defined as the time required to find the solution with n processors divided by the time with one processor. For perfectly parallel implementations, the speedup would be equal to the number of processors. Any communication time and processor synchronization decreases the speedup.

We used a 4×4 grid and varied the number of processors from one to four. While this was a small size problem, it was sufficient to test the parallel implementation. The speedup results are shown in Figure 26. Some speedup can be seen; however, the amount is unsatisfactory.

The low efficiencies indicate that the simplest approach for parallel implementation of the LU-SGS algorithm is inadequate. The results are not surprising since the grain of parallelization in this approach is too fine and requires excessive communication. The number of grid cells in practical applications will be much greater than the number of processors. This suggests dividing the domain into a number of large blocks, so that the grid cells within a block are located on the same processor (and memory) and do not need to communicate through message passing.

4.2 Coarse-Grain Parallelization

In this section, we describe the coarse-grain parallelization of the MHD solver and the performance of this approach applied to a real problem.

The algorithm was parallelized using the domain decomposition technique (DDT). This technique is based on the simple idea of "divide and conquer" The integral form of a general conservation law is

$$\frac{\partial}{\partial t}\int_{\Omega} dV\ Q + \oint_{\Sigma} dS \cdot F(Q) = \int_{\Omega} dV\ S(Q), \tag{60}$$

where Ω is the domain and Σ is the boundary of Ω. Q is the vector of conserved variables, $F(Q)$ is the flux of the conserved variables, and $S(Q)$ is the vector of source terms. By splitting the domain Ω into p subdomains such that

$$\Omega = \bigcup_{i=1}^{p} \Omega_i, \tag{61}$$

one can replace eqn(60) with a set of p conservation equations applied on the subdomains Ω_i.

$$\frac{\partial}{\partial t}\int_{\Omega_i} dV\ Q + \oint_{\Sigma_i} dS \cdot F(Q) = \int_{\Omega_i} dV\ S(Q), \quad i = 1, 2, ..., p \tag{62}$$

Each of these discretized equations is solved by a single processor. Each processor uses the boundary values copied from neighboring subdomains.

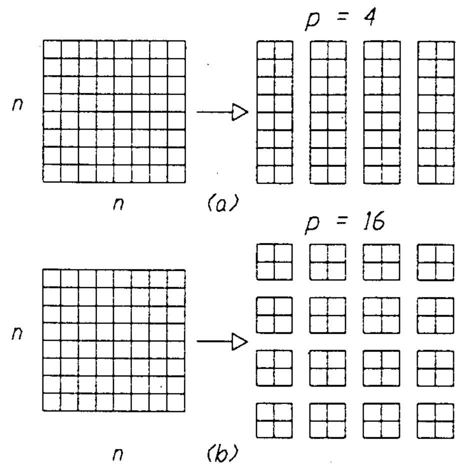


Figure 27: (a) Strip decomposition and (b) patch decomposition of a 2-D domain.

4.2.1 Domain Decomposition

To abstract the computer architecture, we assume that a set of p processors can be assigned to run the code and that these processors implement a message passing system. For simplicity the original domain is assumed to be a square of size $n \times n$.

The 2-D version of the algorithm was ; ırallelized. There are two techniques available for the decomposition of 2-D domains, the strip decomposition and the patch decomposition which are shown in Figure 27.

Strip decomposition is implemented by dividing the original domain in subdomains of $n \times \frac{n}{p}$, and it might be thought of as a 1-D decomposition. With strip decomposition each subdomain needs to exchange data with two neighbors except the subdomains at the boundaries of the original domain which communicate with only one neighbor.

The communication time for an interior subdomain was defined by Zhu[17] as

$$T_{D_{21}} = 2(\sigma + 8\beta n) \tag{63}$$

where σ is the communication start-up time, β is the time required to send one byte of data, and the 8 means that the data are represented as double

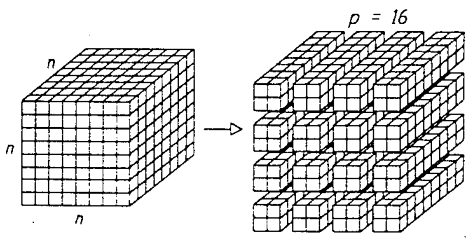


Figure 28: Column decomposition of a 3-D domain is an immediate extension of the patch decomposition of 2-D domains.

precision variables (their size is eight bytes).

Patch decomposition is implemented by dividing the original domain in $\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$. With this method each processor has to communicate with four neighbors unless it is situated on the boundaries of the original domain and it has two or three neighbors. For simplicity it is assumed that p is an even square number and n is evenly divisible by $\sqrt{p}$. The communication time for an internal subdomain is

$$T_{D_{22}} = 4(\sigma + 8\beta\frac{n}{\sqrt{p}}). \qquad (64)$$

For a fixed grid size, $T_{D_{22}}$ decreases with the number of processors since in eqn (64) the number of processors appear at the numerator. In contrast, $T_{D_{21}}$ stays constant with the number of processors.

This made the patch decomposition an obvious choice for our implementation. The technique will also provide a straightforward extension to the column decomposition of 3-D domains (see Figure 28).

4.2.2 Implementation of the Patch Decomposition

The programming model used for the implementation was single program multiple data (SPMD). Each processor runs the same code on the data corresponding to its subdomain. One processor has to perform the domain decomposition and send the data to the other processors. This processor was designated as the *main task*.

Assuming that there are p processors available for running the code they can be arranged in a processor grid of $p_r \times p_c = p$ where p_r is the number of rows and p_c is the number of columns. The size of the original computational domain is $m \times n$. It is possible to have subdomains of equal sizes only if m and n are evenly divisible with p_c and p_r respectively. The domain decomposition was implemented such that some processors receive an extra row or extra column if m and n are not divisible by p_c and p_r.

Physical coupling of the subdomains is accomplished by the exchange of internal boundaries. A processor sends the data from the cells next to its boundaries to the neighboring processors if they exist. The receiving processor assigns the received data to the cells of its respective boundaries. If a processor does not have a neighbor in a certain direction the boundary conditions are applied to that boundary. Since the algorithm uses a five-point stencil only one row/column needs to be exchanged.

4.2.3 Message Passing

One of the goals of the project is to develop a portable code. A first step in assuring the portability was to use a message passing system commonly available on parallel supercomputers and on workstation clusters. This system is the Message Passing Interface (MPI),[18] which was adopted as a standard in May 1994 by industry and academia. Hardware and software vendors' implementation of MPI provides parallel program developers with a consistent set of subroutines callable from FORTRAN77 and C. In our code we made use of the basic point-to-point communications subroutines and global communications subroutines. The point-to-point communication subroutines were used for the domain decomposition and boundary exchange while the global communication subroutines were used for convergence checking. All message passing systems (PVM, MPL) support point-to-point and global communications subroutines so that by using only the basic set we provided for a facile portability to systems not supporting MPI.

4.2.4 Load Balancing

The load balancing for this code is performed by distributing an approximately equal number of cells to each processor. This is accomplished by the *main task* during the domain decomposition phase. Since the number of floating point operations performed by each processor is the same, a static domain decomposition is sufficient to ensure that the processors have an equal share of the computing load. If the code takes were to allow for time-dependent ionization or other localized phenomena which require additional

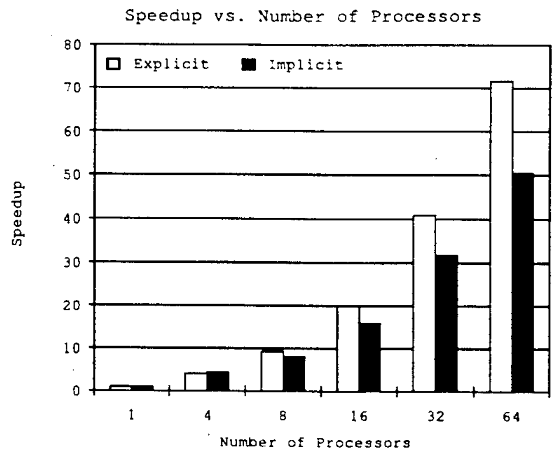


Figure 29: Fixed grid (400×80) speedup results. Note the superlinear speedup of the explicit mode.

operations in a limited region of the computational domain, then a dynamic load balancing procedure may be necessary. A simple algorithm for dynamic load balancing is the masked multiblock described by Borrelli et al.[19] We will implement the masking algorithm in future versions of the code if it becomes necessary.

4.2. Results

In order to measure the performance of the code we applied the parallel version to the plasma gun problem described in Section 3.3. The parallel version was checked against the sequential version, and both produced identical results.

There are two criteria generally used for the performance analysis of parallel codes: (1) the speedup $S_p = T_{seq}/T_p$ and (2) the efficiency $E_p = S_p/p$, where T_{seq} is the time needed for the best sequential algorithm to complete the task and T_p is the time needed for the parallel algorithm run on a number of p processors to complete the same task. Note that the definition of speedup used here is more rigorous and meaningful than the one commonly used since it is based on the sequential version and not the parallel version on one processor.

We ran the parallel code on the IBM SP2 with a fixed grid of size 400×80 on a processor pool of varying size: 4, 8, 16, 32 and 64 processors. The speedup for the explicit and implicit modes is shown in Figure 29. As expected the speedup increases with the number of processors assigned to run the code. For the explicit mode the speedup is superlinear, which seems to contradict Amdahl's law

$$S_p = \frac{T_{seq}}{T_{communication} + \frac{\sum_{i=1}^{p} T_{computation,i}}{p}}. \tag{65}$$

Assuming that no time is used for communication and that the sum of the computation time for all processors is equal to the sequential computation time, the maximum speedup is linear (for p processors the speedup is p). However, Amdahl's law does not take in consideration the architecture of the system used, in particular the cache effects. On the IBM RS/6000 machines, which constitute the nodes of the SP2, the data is passed from the main memory to the CPU through a data cache. A data cache miss involves a delay of eight CPU cycles while the data in the cache can be accessed in one cycle.[20] Noting that an add and multiply operation (a FLOP) takes one CPU cycle the conclusion is that a data cache miss decreases the performance significantly. By increasing the number of processors in the pool and keeping the overall problem size constant, we reduced the amount of data assigned to a processor. Its data cache could hold more data thus reducing the number of cache misses and improving the performance, which explains the superlinear speedup. The same behavior was observed by Michl et al., on a cluster of IBM RS/6000/500 workstations.[21]

The speedup for the explicit mode is higher than that for the implicit mode because the implicit mode is the more computationaly intensive and is, therefore, less sensitive to cache misses. One has to be careful when comparing the results presented in Figure 29 since the number of iterations until convergence is reached for the implicit mode depends on the number of processors used.

The trend of the speedup shows an increasing slope for both explicit and implicit modes which indicates that the code is far from communication saturation. Saturation occurs when the time spent on communications becomes comparable with the computation time. If the number of processors is increased and the size of the subdomains becomes smaller, each processor will have fewer computations to perform, but the total time spent in exchanging the data on the boundaries will increase. The total time spent for boundary exchange can be found using the formula for the communication time for an internal subdomain [eqn(64)] and multiplying it with the number of processors in a pool p,

$$T_{bdry\,exch} = pT_{D_{22}} = 4(\sigma p + 8\beta n\sqrt{p}). \tag{66}$$

The total time spent on boundary exchanges varies proportionally with p.

For the processor pools with a non-square number of processors we have run the code on grids organized as $p_r \times p_c$ and the transpose $p_c \times p_r$, so that the number of row cells versus column cells changed. The results showed that a decomposition whose subdomains have longer rows performs better than one with longer columns. This is consistent with the data cache misses that were observed previously. An improvement of 20–30% in the measured speedup was obtained by modifying the domain partitions. It should be noted that this result is particular to IBM architecture, and the dependency of the obtained speedup on domain decomposition will vary on other architectures. The speedup results shown in Figure 29 for 8 and 32 processors have been averaged.

In order to eliminate the cache effects from the performance analysis we ran the code on grids that scaled with the number of processors. The size of the grid on each processor remained constant. As the number of processors was increased, the grid increased proportionly. The speedup results are presented in Figure 30. Again note that the speedup is measured relative to the sequential version of the code and not the parallel version run on a single processor.

The speedup for a perfectly parallel code for the scaled grid is unity for any number of processors. Our results show a speedup that is less than unity and it decreases with the number of processors. This is an expected result since the total communication time increases with the number of processors. Since the slope of the speedup is gradual and it appears to flatten, we conclude that the parallel code performs satisfactorily on scaled grids.

5 Professional Interactions

5.1 Project Personnel

The personnel directly involved in this project are listed below.

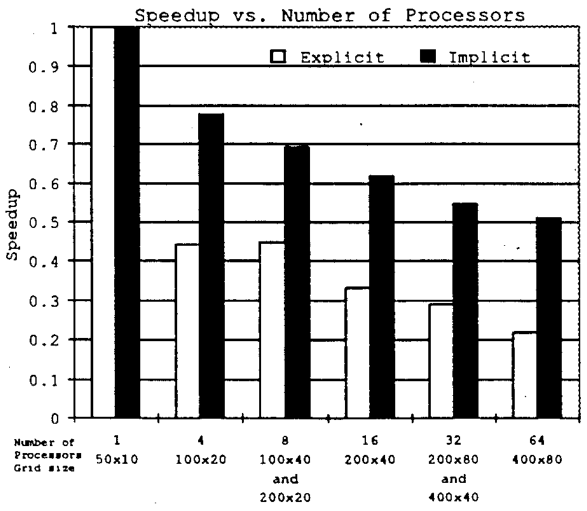


Figure 30: Scaled grid (50×10 per processor) speedup results.

Name	Position
Uri Shumlak	Research Assistant Professor
D. Scott Eberhardt	Associate Professor
Thomas R. Jarboe	Professor
Byoungsoo Kim	Research Associate
Ogden S. Jones	Graduate Student
Bogdan Udrea	Graduate Student
David Taflin	Graduate Student

5.2 Collaborations

5.2.1 Air Force Phillips Laboratory

Dr. Robert Peterkin and Dr. Thomas Hussey of the High Energy Plasma Physics Division on parallelization approaches to MACH3 and on the 3-D Rayleigh-Taylor instability in solid liners. Additional discussions have included liner stabilization by a sheared axial flow.

5.2.2 Lawrence Livermore National Laboratory

Dr. Charles Hartman of the Magnetized Plasmas Division on stabilization of the z-pinch using sheared axial flows. This collaboration resulted in the publication listed below.

5.2.3 University of Michigan

Prof. Bram van Leer, Prof. Kenneth Powell, and Prof. Philip Roe of the Aerospace Engineering Department on approximate Riemann solvers for the MHD plasma model and the zero eigenvalue issue.

5.2.4 University of Colorado

Prof. Steve McCormick of the Applied Math Department on three-dimensional multigrid algorithms.

5.2.5 University of Washington

Prof. Randy LeVeque of the Applied Math Department on approximate Riemann solvers and their applications to multidimensional problems.

5.3 Publications

A journal article has been submitted to the *Journal of Computational Physics*. The title is "An Implicit Approximate Riemann Solver for Non-Ideal Magnetohydrodynamics" by O. S. Jones, U. Shumlak, and D. S. Eberhardt. It has been accepted pending revisions.

A journal article resulting from the collaboration with the Air Force Phillips Laboratory and Lawrence Livermore National Laboratory was published. The article is titled "Sheared Flow Stabilization of the $m = 1$ Kink Mode in Z-Pinches" by U. Shumlak and C. W. Hartman. The citation is Physical Review Letters **75** (18), 3285 (1995).

5.4 Presentations

A paper explaining our project was presented at the Twenty-First Annual IEEE International Conference on Plasma Sciences, Santa Fe, New Mexico, June 1994. The title was "An Implicit Algorithm for the Ideal MHD Equations."

A paper describing our completed two-dimensional parallel code was presented at the Seventh Joint EPS - APS International Conference on Physics Computing, Pittsburgh, Pennsylvania, June 1995. The title was "An Implicit Approximate Riemann Solver for Multi-Dimensional MHD Computations on Parallel Computers."

A paper discussing the magnetic reconnection results was presented at the Thirty-Seventh Annual American Physical Society Meeting of the Division of Plasma Physics, Louisville, Kentucky, November 1995. "Time-Dependent Calculations of Resistive Tearing Instabilities Using a New Implicit MHD Solver."

A paper presenting the findings of the stabilization of the z-pinch by sheared axial flows was presented at the Thirty-Seventh Annual American Physical Society Meeting of the Division of Plasma Physics, Louisville, Kentucky, November 1995. "Sheared Flow Stabilization of the $m = 1$ Kink Mode in Z-Pinches."

6 Conclusions

The successful development of the two-dimensional, viscous, resistive version of the advanced implicit algorithm and the implementation of the algorithm on parallel architectures indicate that we are making significant progress toward our project objectives. This research project has been presented at an international conference, and more presentations are planned. A journal article has been submitted to a refereed journal and its acceptance seems likely.

Valuable collaborations have been formed with the Air Force Phillips Laboratory, Lawrence Livermore National Laboratory, and other universities.

The continuing development of this project will include extending the algorithm to three dimensions, installing the three dimensional algorithm into Phillips Laboratory's MHD code, MACH3, and applying the code to plasma experiments to calibrate the code and gain physical insight.

References

[1] U. Shumlak, T. W. Hussey, and R. E. Peterkin, Jr., IEEE Transaction on Plasma Science **23**, (1995).

[2] B. A. Nelson, T. R. Jarboe, D. J. Orvis, L. McCullough, J. Xie, C. Zhang, and L. Zhou, Phys. Rev. Lett. **72**, 3666 (1994).

[3] F. M. Lehr, A. Alaniz, J. D. Beason, L. C. Carswell, J. H. Degnan, J. F. Crawford, S. E. Englert, T. J. Englert, J. M. Gahl, J. H. Holmes, T. W. Hussey, G. F. Kiuttu, B. W. Mullins, R. E. Peterkin, Jr., N. F. Roderick, P. J. Turchi, and J. D. Graham, J. Appl. Phys. **75**, 3769 (1994).

[4] H. P. Furth, J. Kileen, and M. N. Rosenbluth, Phys. Fluids **6**, 479 (1963).

[5] H. Ok and D. S. Eberhardt, "Solution of Unsteady Incompressible Navier-Stokes Equations Using an LU Decomposition Scheme," AIAA-91-1611 (1991).

[6] S. Yoon and A. Jameson, AIAA J. **26**, 1025 (1988).

[7] P. L. Roe, J. Comp. Phys. **43**, 357 (1981).

[8] R. J. LeVeque, *Numerical Methods for Conservation Laws,* Birkhauser Verlag, Boston (1992).

[9] M. Brio and C. C. Wu, J. Comp. Phys. **75**, 400 (1988).

[10] A. L. Zachery and P. Colella, J. Comp. Phys. **99**, 341 (1992).

[11] K. G. Powell, B. van Leer, and P. L. Roe, *Private Communication,* 1994.

[12] H. P. Furth, J. Killeen, and M. N. Rosenbluth, Phys. Fluids **6**(4), 459 (1963).

[13] H. P. Furth, Phys. Fluids **28**(6), 1595 (1985).

[14] J. Killeen and A. I. Shestkov, Phys. Fluids **21**(10), 1746 (1978).

[15] D. Schnack and J. Killeen, J. of Comp. Physics **35**, 110 (1980).

[16] Al Geist, A. Beguelin, J. Dongara, W. Jiang, R. Manchek, V. Sunderam, *PVM 3 User's Guide and Reference Manual,* 1994.

[17] J. Zhu, *On the Implementation Issues of Domain Decomposition Algorithms for Parallel Computers* Parallel CFD Conference, 1992.

[18] Message Passing Interface Forum *MPI: A Message Passing Interface Standard,* May 5, 1994.

[19] S. Borrelli, A. Matrone, P. Schiano, *A Multiblock Hypersonic Flow Solver For Massively Parallel Computers,* Parallel CFD Conference, 1992.

[20] IBM Corp., *Optimization/Tuning Guide for XL FORTRAN and XLC.*

[21] T. Michl, S. Wagner, M. Lenke, A. Bode, *Dataparallel Implicit Navier-Stokes Solver on Different Multiprocessors* Parallel CFD Conference 1993.